Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 8, No. 1, February, Pp. 461-472

Sensor fusion using model predictive control for differential dual wheeled robot

Achmad Imam Sudianto*[1], Muhammad Aziz Muslim[1], Moch Rusli[1]
Brawijaya University, Indonesia[1]

Article Info
Keywords:
LIDAR, Camera, Artificial Neural Network, Model Predictive Control, Convolutional Neural Network, Mobile Robot

Article history:
Received: December 12, 2022
Accepted: January 12, 2023
Published: February 28, 2023

Cite:
A. I. Sudianto, M. A. Muslim, and M. Rusli, "Sensor Fusion using Model Predictive Control for Differential Dual Wheeled Robot ", KINETIK, vol. 8, no. 1, Feb. 2023. https://doi.org/10.22219/kinetik.v8i1.1614

*Corresponding author.
Achmad Imam Sudianto
E-mail address:
imamsudianto@student.ub.ac.id

Abstract
Every mobile robot mission starts with the robot being moved to the task site. From there, the robot executes its tasks. A control system is required to move the mobile robot's actuator (which may be in the shape of wheels or legs) and comprehend the environment around the robot to perform these movements (perception). This research aims to develop a technique to control a robot's movement while detecting obstacles and distances toward an object. The robot is equipped with LIDAR and a camera to perform these tasks. The control is divided into two major parts, low-level and high-level controller. As part of a low-level controller robot, the Model Predictive Control (MPC) method is proposed to help with the control of wheel while the Artificial Neural Network (ANN) approach to use in this study to identify obstacles and the Convolutional Neural Network (CNN) method for detecting objects, both ANN and CNN as control for high-level part of the robot. The results of this study can prove that CNN can help detect existing objects with a value 45% for detecting an object. The obtained result from the MPC method, which has been combined with a ANN as an obstacle detector, is that the smaller the horizon value, the shorter the time needed to reach the desired coordinates with the result being 45 seconds.

1. Introduction

Autonomous vehicle research has grown significantly over the last few decades. The Unmanned Ground Vehicle is an illustration of an autonomous vehicle (UGV). Path tracking and path following are two examples of how it carries out its mission. UGVs are furnished with a number of parts, including a frame, motor, controller, and battery[1]. Operating a UGV requires complex control and navigation procedures[2]. When using its sensors to navigate autonomously, the UGV's location must always be known. LIDAR (Light Detection Ranging), IMU (Inertia Measurement Unit), GPS (Global Positioning System), UWB (UltraWide Band), and other sensors are some of the sensors utilized to enable the navigation of a UGV[3]. LIDAR, one of these sensors' features, is utilized to identify impediments while the IMU is used to provide information on speed, orientation, and gravity using a combination of accelerometer, gyroscope, and magnetometer.

Researchers have suggested a number of navigational techniques. Lane Keep Assist (LKA) and collision avoidance are two components of the navigation system known as Advanced Driving Assistance Systems (ADAS)[4]. The LKA feature is required to prevent the car from veering off course. On the other hand, if there are obstructions in the way that the vehicle is traveling, collision avoidance is a feature to avoid [5]. The vehicle can avoid collisions and crashes thanks to the collision avoidance sensor feature. Contrarily, the LKA requires a low-level controller with strong capabilities in order for the motor to follow the path indicated by the robot navigation system [6]. In other studies, one of the methods that can be used as collision avoidance is the Artificial Neural Network (ANN) method, where this method can be used as a support for the navigation system of a mobile robot, of course this cannot be separated from the use of qualified sensors such as LIDAR sensors [7].

In other works of literature, numerous investigations have investigated various control strategy theories and methods. For instance, research that integrated the PID (Proportional Integral Derivative), LQR (Linear Quadratic Controller), and MPC (Mode Predictive Control) control techniques discovered the benefits of the MPC method over others [8]. The peak amplitude obtained with MPC has a shorter time, according to other studies, and MPC is preferable to PID and LQC (Linear Quadratic Controller) controllers in terms of optimizing DC motor speed [9]. The three control methods have different settling time outcomes [10]. Convolutional Neural Networks (CNN)-based machine learning has emerged as the industry standard for a variety of computer vision tasks, including those carried out by mobile robots[11]. One deep learning method for identifying content images is CNN, which has shown good performance in image segmentation, classification, detection, and retrieval related tasks [12] The performance of the CNN method can also be seen in YOLO (You Only Look Once) [13].

In this research, the UGV will be represented by a mobile robot that described in the research methods section. The UGV will rely on LIDAR and camera for perception as the primary sensor. The main weakness of the camera is its unreliability in dealing with changes in lighting, as well as the low accuracy for estimating the distance which is the main requirement in carrying out movements. Combining information from several types of sensors is expected to overcome these weaknesses. To carry out this movement, a control method is needed to move the mobile robot actuator (in the form of wheels). By combining the results of previous studies, such as the superiority of MPC compared to PID or LQR, a neural network that can be used as a collision avoidance method and CNN that can detect and recognize objects around it, this research will develop an MPC method to regulate the movement of mobile robots based on LIDAR sensor and camera using machine learning-based artificial intelligence techniques.

2. Research Method

In this research, it is broadly divided into two sub-chapters which here will explain the use of the proposed methods such as ANN, CNN and MPC control. This section also describes the outline of the research to be carried out.

2.1 Outline of the Research

According to the caption of Figure 1, the robot navigation system implemented in this research consists of three key stages: the neural network stage, the YOLO stage, and the control technique (MPC) stage. The process of creating a neural network first gets the datasets ready for processing and training. On the CNN section, this following section is use YOLO weights to detect objects, and the last step is to use the MPC method to control the motors. We employ the neural network's data set as a trained weight to move toward the coordinate target, with the output controlling the motor. The MPC method is used to regulate the motor's speed, both linearly and angularly.

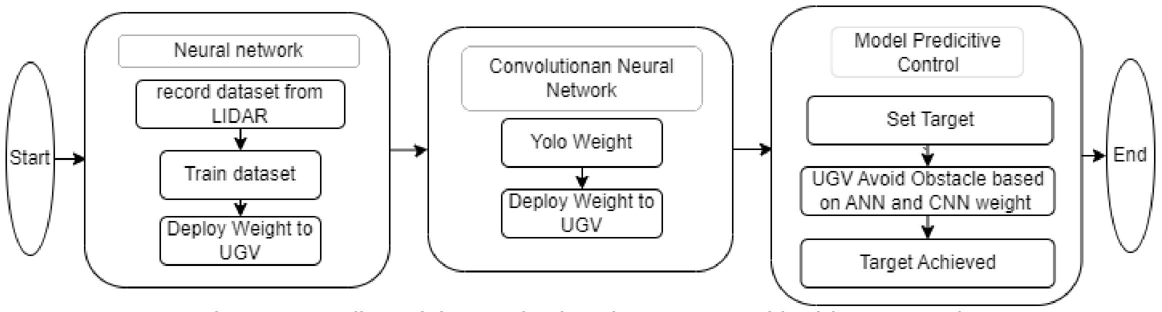
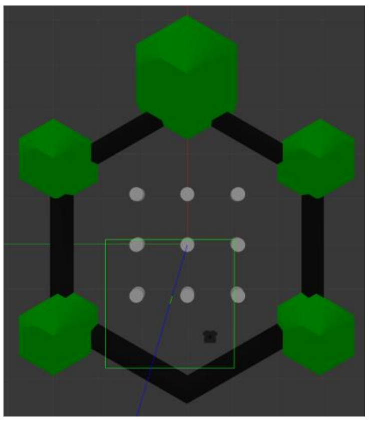

Start
Neural network
record dataset from LIDAR
Train dataset
Deploy Weight to UGV
Convolutionan Neural Network
Yolo Weight
Deploy Weight to UGV
Model Predicitive Control
Set Target
UGV Avoid Obstacle based on ANN and CNN weight
Target Achieved
End

Figure 1. Outline of the Method and Stages used in this Research

2.2 RVIZ and Gazebo

The Gazebo simulator, created by Open Robotics, was utilized in this study to help algorithm development for mobile robotics[14]. To establish the concept and illustrate the results of sensors, we used RVIZ. The simulation was distributed-run, and Gazebo was used to give planning and show how necessary sensors. The simulation used the Turtlebot mobile robot model as a UGV depicted in Figure 2 section (a), the figure also shows the location of the obstacles and the environmental model in the simulation. In Figure 2 in section (b), the LIDAR and camera components used in this UGV are also shown.


(a)

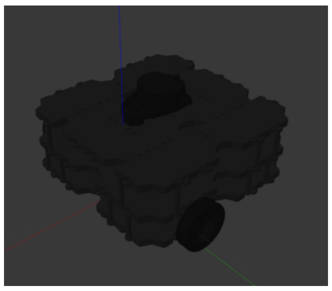
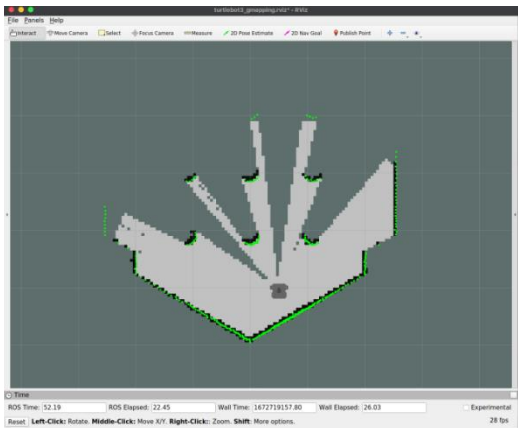

(b)

Figure 2. (a) Environmental Model in Gazebo and (b)Turtlebot Robot as UGV

Distance estimation is typically done using LIDAR [15]. It is made up of a laser, a rotating mirror that scans the surrounding area with the laser beam, and a detector that calculates an object's distance based on the laser's time-of-flight. In this research, the Turtlebot LIDAR module is used. It has a maximum range of 3.5 meter, a resolution of 0.5 degrees, and a scanning speed of 300 ± 10 rpm (revolutions per minute). It is mounted on the top of the UGV, allowing objects to stick out vertically above it. The visual displayed on RVIZ is in the form of LIDAR scan results, where the results are obtained from scanning the environment around the robot with an area of 360 degrees. Later, the values that will be processed by the method is the raw data in the form of numbers, not image forms as shown in RVIZ. Figure 3 illustrates how the RVIZ program presents the LIDAR scanning results.



ROS Time: 52.19 ROS Elapsed: 22.45 Wall Time: 1672719157.80 Wall Elapsed: 26.03

Left-Click: Rotate. **Middle-Click:** Move X/Y. **Right-Click:**: Zoom. **Shift**: More options.

Figure 3. LIDAR Scanning Result in RVIZ

2.3 Kinematic Model of UGV

The UGV's motion model is represented in Figure 4. It is made up of a vehicle frame, two drive wheels, and an axle-mounted front sliding blade. The UGV may change direction and speed by using its two driving wheels, each of which is powered by a separate motor. A vital part of a motion controller is a low-level MPC controller that regulates a UGV's motion [16].

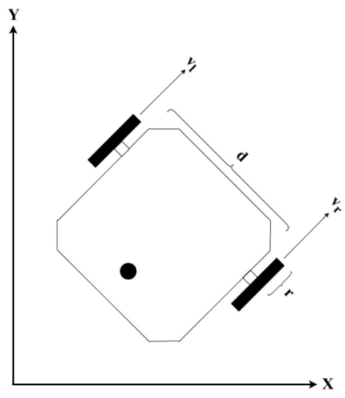


Figure 4. Kinematic Model of UGV

Wheel speed in Equation 1 and the formula of each velocity right and left in Equation 2.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}cos(\theta) & \frac{r}{2}cos(\theta) \\ \frac{r}{2}sin(\theta) & \frac{r}{2}cos(\theta) \\ \frac{-r}{2d} & \frac{-r}{2d} \end{bmatrix} \begin{bmatrix} v_l \\ v_r \end{bmatrix} \tag{1}$$

$$v_r = rw_r \text{ and } v_l = rw_l \tag{2}$$

Vehicle speed and heading rate equation.

$$v = \frac{r}{2}(v_r + v_l)$$
$$w = \frac{r}{2d}(v_r - v_l) \tag{3}$$

Together with the robot axis, v represents the point up's linear velocity, and w represents the angle's angular velocity (Equation 3). The kinematics Equation 4 above therefore has the following definition.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} cos(\theta) & 0 \\ sin(\theta) & 0) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} \tag{4}$$

Equation 4 represents the UGV's steering system. The main objective is the intended reference trajectory, which may control the system's course. The control algorithms are designed to generate sufficient left and right wheel speeds to move the mobile robot along the needed direction trajectories. It is intended that all of the vehicle's attributes will be measured using x, y, and θ. Therefore, the vehicle may be entirely controlled by the two inputs, v and w (the vehicle's rotational velocity and linear velocity).

2.4 Artificial Neural Network (ANN)

ANN is used as a method to train data obtained through LIDAR [16]. The data was gathered using LIDAR, which the robot uses for its perception system. The data obtained is in the form of a value for each degree where each degree has a value for the distance between the LIDAR and the obstacle (described in sub-section 2.2). However, as described in previous sub-section the data that is processed is in the form of numbers, not in the form of images. The raw data that obtained from LIDAR scanning results will be processed by a three-layer classifier neural network. The neural network has three layers with 40 nodes in input layer (IL) 100 nodes in hidden layer (HL) and 2 nodes in output layer (OL). Proposed neural network architecture is shown in Figure 5.

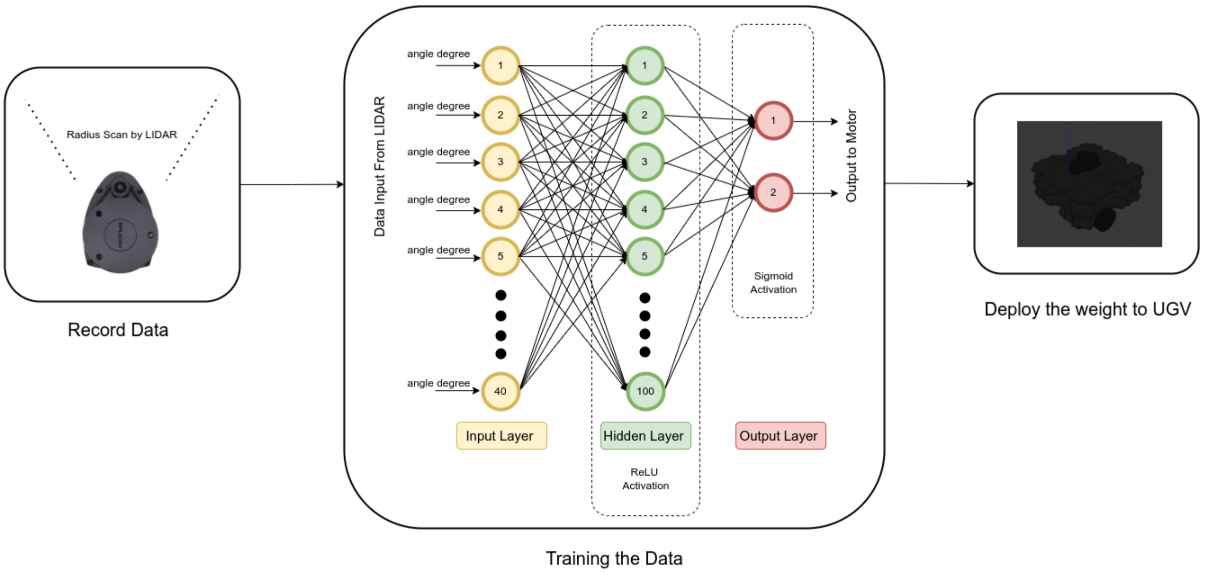


Figure 5. Neural Network Method Flow and Architecture

Experimentally determined hidden layer node counts are used. If no obstacles are found within 3.5 meter, the 40 neurons' inputs are ordered to "infinity". Activation function is needed to activate each node, the activation function used in this research is Relu (Rectified Linear Units) and sigmoid function [17]. Two neurons in the Output Layer (OL), which determine the robot's velocity and steering angle, are activated using the sigmoid function. Relu are activated in 100 neurons in the Hidden Layer (HL) according to Equation 5.

$$f(x) = \begin{cases} 0, for\ x\ <\ 0 \\ x, otherwise \end{cases} \quad (5)$$

The neural network is trained to navigate by describing approximately 412 parameters. So on at the end of the layer there is an activation function in the form of a sigmoid function as in Equation 6.

$$f(x) = \frac{1}{1\ +\ e^{-x}} \quad (6)$$

Where the value of this section will be continued to the controller section on the MPC. The output we will get is in accordance with the following Equation 7 using the sigmoid function.

$$output = \sigma(\sum_{i=0}^{n} x_i w_i\ +\ b_i) \quad (7)$$

Where x_i , w_i , b_i and n are the inputs, weight vector, bias and number of neurons, respectively. To evaluate and measure of success used in the ANN method is an evaluation matrix, where the matrix has several parameters [17], namely accuracy (Equation 8), precision (Equation 9), recall (Equation 10), and F1-score (Equation 11). A TP is an outcome where the classification algorithm accurately predicted the positive label. Similarly, if the algorithm properly predicts the negative label, a TN is the outcome of the classification. When the classifier predicts a negative class as positive, this is known as FP. The forecast of a positive label as a negative is represented by the final confusion matrix term, FN. The harmonic means of the precision and recall values is acquired in order to determine the proposed model's accuracy, the F1-score is calculated using the formula shown in equation (11) to determine the reliability of the model.

$$Accuracy\ = \frac{TN + TP}{TN + TP + FN + FP} \quad (8)$$

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall\ = \frac{TP}{TP + FN} \quad (10)$$

$$F1 - Score = \frac{2\ *\ Precision\ *\ Recall}{Precision\ +\ Recall} \quad (11)$$

2.5 You Only Look Once (YOLO)

A single-stage object detection system called YOLO [18] is based on Deep Convolutional Neural Networks (DCNNs). A backbone layer and a detection layer make up YOLO. The output feature map is created by extracting features from the backbone layer [19]. Darknet is deployed as the backbone for the system. The detecting procedure will proceed on the output feature map.

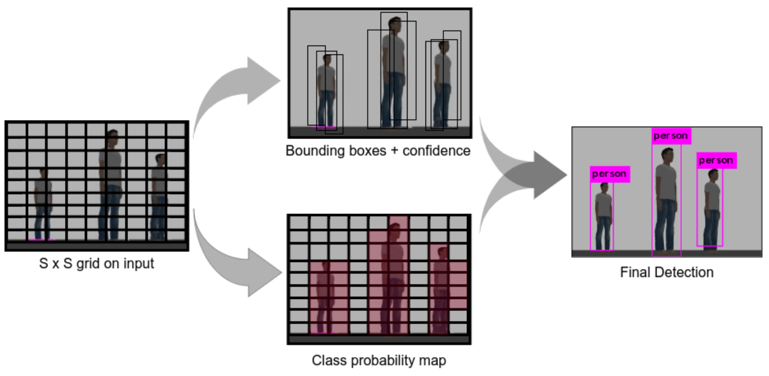


Figure 6. Cross Probability Map

Figure 6 explains the detection method in detail. YOLO receives input photos that are mapped into SxS-sized grid cells. The bounding box must be predicted for each grid cell with a confidence score and class probability [20]. Five parameters x, y, w, h, confidence, and class probability—as well as Class probability prediction are included for each predicted box. In the representation, w and h stand for the bounding box's width and height, x and y for the center's coordinates, and C for the conditional class probability. The explanation of Class Probability Representation is given in Equation 12.

$$\boldsymbol{Class\,probability} = \boldsymbol{Pr}\,(\boldsymbol{Classi} \mid \boldsymbol{Object}) \tag{12}$$

The conditional class probabilities in Equation 12 are conditioned on grid cells containing objects, and the confidence score reflects how sure the bounding box contains objects, and how accurate the bounding box is according to estimates [8]. The confidence score is represented Equation 13.

$$\boldsymbol{Confidence\ score} = \boldsymbol{Pr}\,(\boldsymbol{Object}) * \boldsymbol{IoU}\,(\boldsymbol{truth\,predict}) \tag{13}$$

In (9), Pr(Object) displays the likelihood that an object will be within the bounding box, and IoU (Intersec over Union) predic displays the intersection of the ground truth box and prediction box (IoU). IoU has a susceptible value between 0 and 1. If the IoU value is close to 1, it means that the bounding box prediction is accurate and close to the ground truth value. If the IoU value above the threshold, then the prediction of the bounding box containing an object is correct. IoU is frequently used to assess the accuracy of the object detection from the set threshold value. The IoU threshold value is often set by default to be 0.3. To find out how the performance of the hardware used, measurements can be made with FPS (Frame per Second), here is a equation to find out FPS Equation 14).

$$\boldsymbol{FPS} = \boldsymbol{Number\ of\ Frame} \,/\, \boldsymbol{Processing\ time\ in\ second} \tag{14}$$

2.6 MPC

Model predictive control (MPC), is a control technique, to try to predict the system's behavior across a finite time window, called the horizon[21]. The optimal control inputs with regard to a specified control target and subject to system restrictions are determined based on these predictions and the present measured/estimated state of the system. The procedure of measuring, estimating, and computing is repeated after a predetermined amount of time with a shifting horizon [22].

To sets the objective of the optimal control problem, introduce the following cost function.

$$\boldsymbol{J(x, u, z)} = \sum_{\boldsymbol{k=0}}^{\boldsymbol{N}} \left(\boldsymbol{l(x_k, z_k, u_k, p_k, p_{tv,k})}\right) + (\Delta \boldsymbol{u}_{\boldsymbol{R}}^{\boldsymbol{T}} \boldsymbol{R} \Delta \boldsymbol{u_k}) + \boldsymbol{m(x_{N+1})} \tag{15}$$

Equation 15 is used to set the lagrange term $(\square(\square_{\square},\square_{\square},\square_{\square},\square_{\square},\square_{\square\square,\square})$(*lterm*) and meyer term $\square(\square_{\square+\square})$ (*mterm*), where N is the prediction horizon [23]. The values of parameter we used in mterm is 0.7 and lterm is 0.3.

When set the penalty factor for the inputs, the Equation 16 and Equation 17 referring to the input names in model and the penalty factor as the respective value. Define for i ∈ II, where I is the set of inputs and all k=0,…,N where N denotes the horizon.

$$\Delta \boldsymbol{u_{k,i}} = \boldsymbol{u_{k,i}} - \boldsymbol{u_{k-1,i}} \tag{16}$$

$$\sum_{\boldsymbol{k=0}}^{\boldsymbol{N}} \sum_{\boldsymbol{i \in I}} \boldsymbol{r_i} \Delta \boldsymbol{u_{k,i}^2} \tag{17}$$

In this research we set rterm, lterm and the weighted squared cost to the MPC as objective function (Equation 15).

3. Results and Discussion

This section presents the results of the experiments conducted. As previously mentioned, the results are divided into several categories, namely how the data training results are presented by ANN, the detection results obtained by CNN, and how the MPC control works.

3.1 Neural Network and Training Result

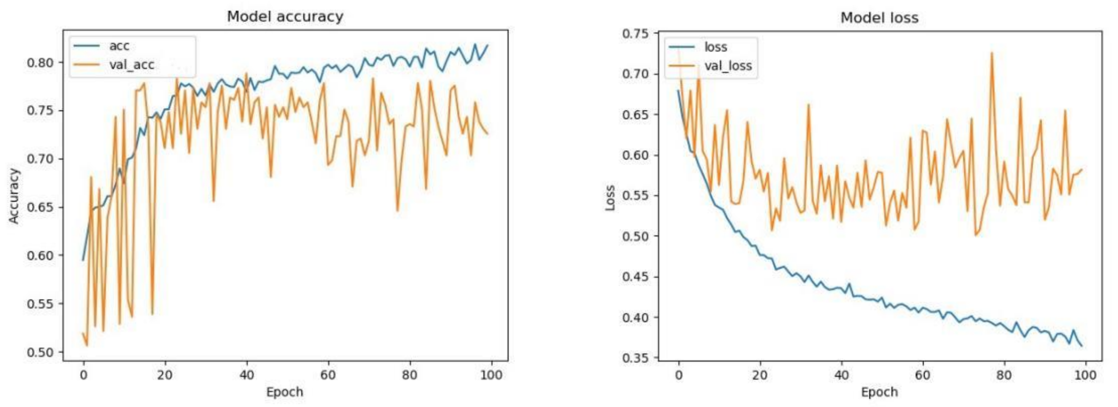
Without human involvement, mobile robots have to adapt their environment. It must be capable of collecting environmental data. So, to scan the working area, we suggest using a LIDAR with a range of 3.5 meters and a field of view of 40 degrees. The environment model changes every time an object identification scan is carried out. Robots can calculate the distances between barriers. The architecture of our algorithm is shown in Figure 5.

Before using the dataset, training and data validation are required. In this research, KFold Cross validation was used as a validation method. Kfold cross validation is used to evaluate the validity of the derived model from the dataset received from LIDAR. The dataset utilized is split into 90% train data and 10% test data on kfold, where the value used for the k variable is 10. Additionally, it shows that 10 iterations of kfold were performed using the dataset obtained from LIDAR. The processed data's results are shown in Table 1.

Table 1. Score of KFold

KFold Cross Validation					
No. Test	**1**	**2**	**3**	**4**	**5**
Value	0.87	0.81	0.83	0.81	0.84
No. Test	**6**	**7**	**8**	**9**	**10**
Value	0.81	0.81	0.89	0.84	0.85

The network is trained and calculate 1000 data, and at various epochs, the efficiency objective is achieved. Figure 7 is the result of the graph obtained through training.



(a) (b)
Figure 7. (a) Model Accuracy and (b) Model Loss Graphic

The neural network training with the following parameters produced the graph above: epochs = 100, batch size = 8, and validation split = 0.2. According to Figure 7, each epochs resulted in reduced loss and increased acc (accuracy), Val_acc and loss start to increase as the model learns, indicating that the model was designed appropriately. Table 2 contains the final score after the training, val_loss and val_acc values for the last epoch are displayed too. The metrics derived from the training and test datasets, respectively, are loss and val_loss. Similar to acc, val_acc refers to the accuracy result on the test dataset. For the most accurate representation of model performance, use val_acc. As a result, 0.73 val_acc value is chosen as the ANN's last performance in the research.

Table 2. Final Score Obtained

Parameter	Value (%)
Acc	0.83
Val_acc	0.73
Loss	0.32
Val_loss	0.58

Several experiments were run to demonstrate the effectiveness of the suggested strategy, and the results were presented using some common matrices like classification accuracy, precision, recall, and F1-score regarding to equation 8, 9, 10, and 11. When making predictions in novel settings, this can help to identify the model's advantages and disadvantages. The model performance of the suggested approach is crucial for machine learning. The findings from the parameters described, are summarized in Table 3.

Table 3. Score Obtained

Parameter	Value (%)
Precision	0.93
Recall	0.84
Accuracy	0.88
F1-Scre	0.88

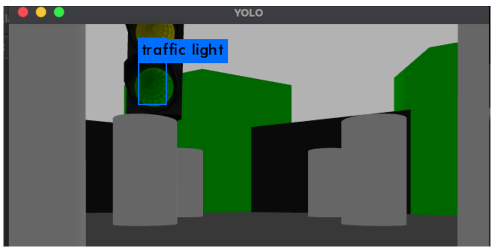
3.2 Convolution Neural Network and Training Result

The use of CNN is to find out the objects that are on the front of the robot, later this information is used to detect the type of the object [24]. The weights used are the weights from the YOLO V2 and YOLO V3 COCO (Common Objects in Context) datasets. The choice of YOLO as the object detection algorithm is because the mAP (mean average prediction) value of this algorithm outperforms several other methods such as SSD and Faster R-CNN [25]. The workings of the YOLO v3 Tiny network architecture used can be seen in Table 4.

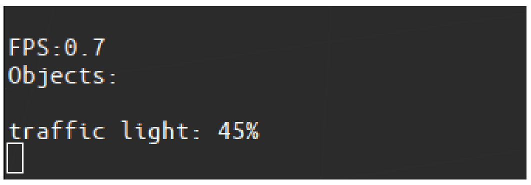
Table 4. YOLO v3 Tiny Network Architecture

Layer	Type	Filter	Size/Stride	Input	Output
0	Convolutional	16	3 x 3 / 1	416 x 416 x 3	416 x 416 x 16
1	Max Pooling		2 x 2 / 2	416 x 416 x 16	208 x 208 x 16
2	Convolutional	32	3 x 3 / 1	208 x 208 x 16	208 x 208 x 32
3	Max Pooling		2 x 2 / 2	208 x 208 x 32	104 x 104 x 32
4	Convolutional	64	3 x 3 / 1	104 x 104 x 32	104 x 104 x 64
5	Max Pooling		2 x 2 / 2	104 x 104 x 64	52 x 52 x 64
6	Convolutional	128	3 x 3 / 1	52 x 52 x 64	52 x 52 x 128
7	Max Pooling		2 x 2 / 2	52 x 52 x 128	26 x 26 x 128
8	Convolutional	256	3 x 3 / 1	26 x 26 x 128	26 x 26 x 256
9	Max Pooling		2 x 2 / 2	26 x 26 x 256	13 x 13 x 256
10	Convolutional	512	3 x 3 / 1	13 x 13 x 256	13 x 13 x 512
11	Max Pooling		2 x 2 / 2	13 x 13 x 512	13 x 13 x 512
12	Convolutional	1024	3 x 3 / 1	13 x 13 x 512	13 x 13 x 1024
13	Convolutional	512	3 x 3 / 1	13 x 13 x 1024	13 x 13 x 512
14	Convolutional	425	3 x 3 / 1	13 x 13 x 512	13 x 13 x 425
15	Detection				

Through the network architecture of YOLO method, the process carried out in this detection is passed through several stages of the convolution layer, namely from 416*416 pixels to 13*13 pixels [26]. The performance of the computer used is bflop/s ranging from 0.150 to 1.595 so this value affects the fps produced when the detection is carried out. In the simulation as shown below, there are 1 object that can be detected. An object is a traffic light that we can see in Figure 8. The percentage of object detection found in the simulation results is 45%.



(a)



```
FPS:0.7
Objects:

traffic light: 45%
```

(b)

Figure 8. (a) Object Detected and (b) FPS Result and Object Detection in Percentage

There are some speed differences when using YOLO, because there are 2 types of YOLO methods used. The basic difference between the two methods is the speed and accuracy when detecting an object. Tiny-yolov3 is a simplified version of YOLOv3, which has a much smaller number of convolution layers than YOLOv3, which means that tiny-yolov3 does not need to occupy a large amount of memory, reducing the need for hardware. And it also greatly speeds up detection, but lost some of the detection accuracy. The differences in fps are presented in the following Table 5.

Table 5. Result of FPS

Method	FPS
YOLO v3 - Tiny	2
YOLO v3	0.7

3.3 Model Predictive Control Result

Signal inputs in the form of waypoint are used to simulate the MPC control system of the autonomous robot [27]. The robot controller must calculate the input control value that has been provided in order to arrive at the waypoint value, which serves as a reference value.

In the autonomous robot system, prediction control is utilized to ensure that the car moves to the predetermined target coordinate [28]. The simulation that results receives numerous values, including the robot's speed and odometry. All initial values on the graph are not 0, as the robot does not begin at the 0.0 coordinate position (see the graphic in Figure 9, 10 and 11). The graph shows that x-coordinate odometry is represented by blue, whereas y-coordinate odometry is shown by red. While the colors green and yellow, respectively, represent linear velocity x and angular velocity z. By changing the value of N, as in equation (15) which is the horizon prediction value, here are some experimental results with various horizon prediction values ranging from 10, 25 to 30.

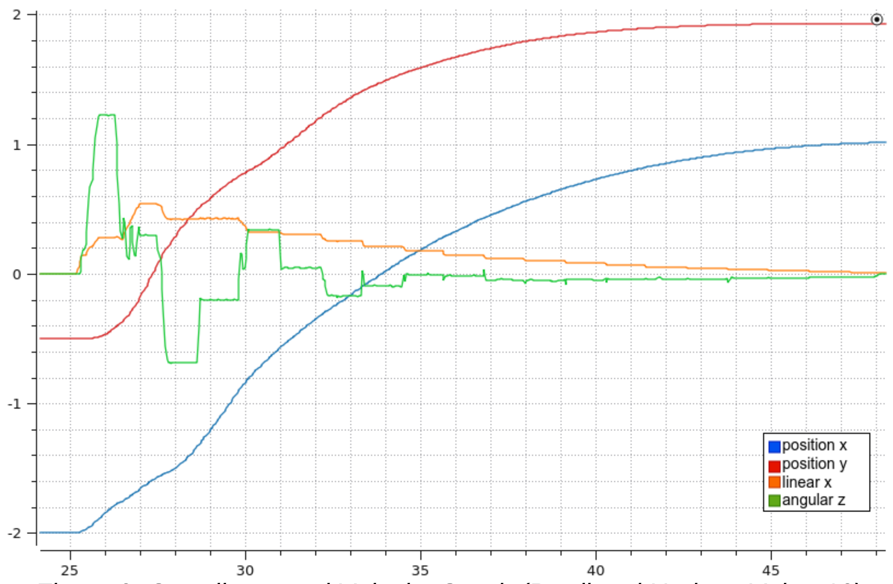


Figure 9. Coordinate and Velocity Graph (Predicted Horizon Value 10)

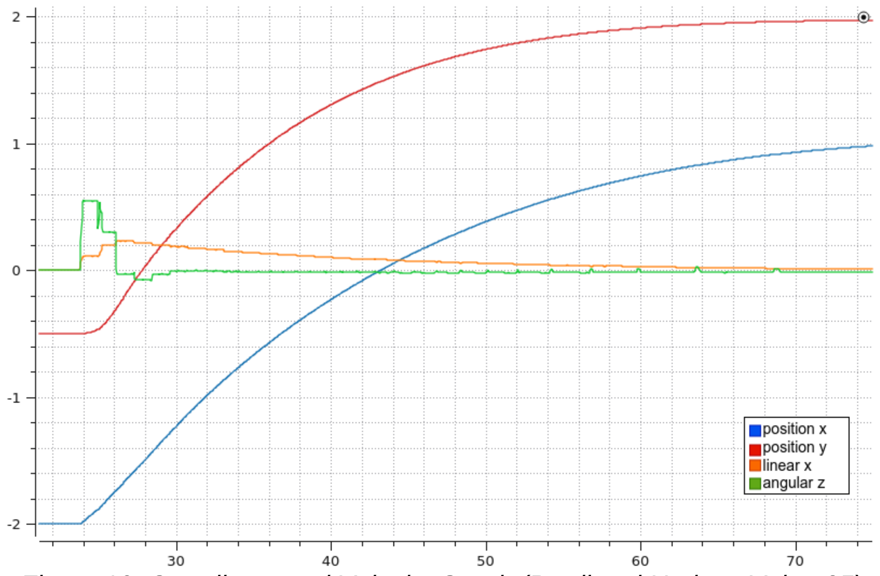


Figure 10. Coordinate and Velocity Graph (Predicted Horizon Value 25)

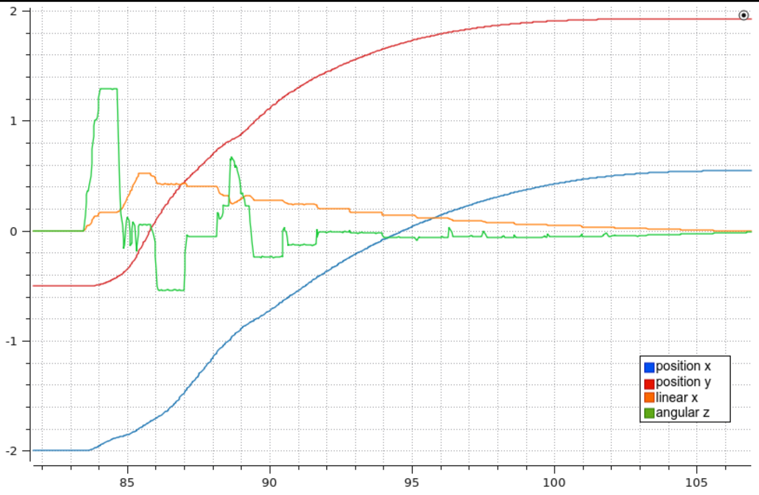


Figure 11. Coordinate and Velocity Graph (Predicted Horizon Value 30)

In order to better validate performance, experiments were run with different horizon predictive value variables according to the values in Table 6, it was proven that these factors had an impact on MPC performance. Figure 9, 10 and 11 show how the robot moves in accordance with predefined coordinates. The robot's beginning position is at x = -2 and y = -0.5, and its final position is at x = 1 and y = 2. The processing time required by a system to arrive at the desired coordinates will increase as the horizon value increases. Systems affected by a horizon value of 30 react slower than systems affected by other horizon values (10 and 25) as shown in Table 6. The problem can be solved by lowering the predicted horizon value. As the horizon prediction value decreases, the control time consumption decreases, as can be seen from the fluctuation of the position versus time graph in Figure 7 to 9. These results are summarized in Table 6.

Table 6. Comparison of Horizon Value and Time of Achievement

Prediction Horizon	Time (sec)
10	47
25	74
30	105

4. Conclusion

This analysis shows how the lateral and longitudinal positions of the robot can be adjusted by modifying the control system, commonly known as the robot's position and speed controller. These two factors are essential for making sure the car keeps moving in a way that will enable it to arrive at a particular coordinate location. Based on this description, the dynamics of the vehicle in three degrees of freedom are represented by the dynamic equations x, y, and theta. In the performance of the training results using the activation function values, the mobile robot obstacle avoidance approach utilizing the ANN algorithm demonstrated an accuracy score of 0.83 and a loss of 0.32 in the results acquired from the above mentioned experiments. Additionally, shortening the MPC's horizon value will enable it to reach the target sooner (45 seconds at a horizon value of 10).

The fundamental weakness in using CNN in this research is the inability of the CPU used in the robot, so that the computation used cannot display the appropriate speed for use in real situations. This can be proven by the results of the fps value, which is only worth 2 FPS.

Notation

□$_{□}$	= the states of the	□$_{□}$	= control inputs	□$_{□}$	= algebraic states
m	= meyer term	x_i	= inputs	b_i	= bias
N	= horizon value	TP	= True Positive	TN	= True Negative
FN	= False Negative	FP	= False Positive	l	= lagrange term
w_i	= weight vector	V_r	= Right wheel speed	v	= Vehicle speed
□$_{□}$	= uncertain parameters	x	= Global vehicle x-position	r	= Wheel radius
□$_{tv,□}$	= time-varying parameters	□$_{□}$	= time-varying measurements	V_l	= Left wheel speed
y	= Global vehicle y-position	θ	= Global vehicle heading	w	= Vehicle heading angular
velocity		w_l	= Angular left wheel velocity	d	= The distance between two
driving wheels					

References

[1] Rafaila, R. C., Caruntu, C. F., & Livint, G. (2016). Centralized model predictive control of autonomous driving vehicles with Lyapunov stability. 2016 20th International Conference on System Theory, Control and Computing, ICSTCC 2016 - Joint Conference of SINTES 20, SACCS 16, SIMSIS 20 - Proceedings, 663–668. https://doi.org/10.1109/ICSTCC.2016.7790742

[2] Peng, Y., Qu, D., Zhong, Y., Xie, S., Luo, J., & Gu, J. (2015). The obstacle detection and obstacle avoidance algorithm based on 2-D lidar. 2015 IEEE International Conference on Information and Automation, ICIA 2015 - In Conjunction with 2015 IEEE International Conference on Automation and Logistics, 1648–1653. https://doi.org/10.1109/ICInfA.2015.7279550

[3] Hutabarat, D., Rivai, M., Purwanto, D., & Hutomo, H. (2019). Lidar-based obstacle avoidance for the autonomous mobile robot. Proceedings of 2019 International Conference on Information and Communication Technology and Systems, ICTS 2019, 197–202. https://doi.org/10.1109/ICTS.2019.8850952

[4] Kaleci, B., Turgut, K., & Dutagaci, H. (2022). 2DLaserNet: A deep learning architecture on 2D laser scans for semantic classification of mobile robot locations. Engineering Science and Technology, an International Journal, 28. https://doi.org/10.1016/j.jestch.2021.06.007

[5] Maddalena, E. T., da Moraes, C. G. S., Waltrich, G., & Jones, C. N. (2020). A neural network architecture to learn explicit MPC controllers from data. IFAC-PapersOnLine, 53(2), 11362–11367. https://doi.org/10.1016/j.ifacol.2020.12.546

[6] Ghorpade, D., Thakare, A. D., & Doiphode, S. (2018, September 11). Obstacle Detection and Avoidance Algorithm for Autonomous Mobile Robot using 2D LiDAR. 2017 International Conference on Computing, Communication, Control and Automation, ICCUBEA 2017. https://doi.org/10.1109/ICCUBEA.2017.8463846

[7] Shalumov, A., Halaly, R., & Tsur, E. E. (2021). LiDAR-driven spiking neural network for collision avoidance in autonomous driving. Bioinspiration and Biomimetics, 16(6). https://doi.org/10.1088/1748-3190/ac290c

[8] Tavernini, D., Metzler, M., Gruber, P., & Sorniotti, A. (2019). Explicit Nonlinear Model Predictive Control for Electric Vehicle Traction Control. IEEE Transactions on Control Systems Technology, 27(4), 1438–1451. https://doi.org/10.1109/TCST.2018.2837097

[9] S. Sahoo, B. Subudhi and G. Panda, "Optimal speed control of DC motor using linear quadratic regulator and model predictive control," 2015 International Conference on Energy, Power and Environment: Towards Sustainable Growth (ICEPE), 2015, pp. 1-5, https://doi.org/10.1109/EPETSG.2015.7510130

[10] D. Tavernini, M. Metzler, P. Gruber and A. Sorniotti, "Explicit Nonlinear Model Predictive Control for Electric Vehicle Traction Control," in IEEE Transactions on Control Systems Technology, vol. 27, no. 4, pp. 1438-1451, July 2019, https://doi.org/10.1109/TCST.2018.2837097

[11] Giusti A, Cireşan D C, Masci J, Gambardella L M and Schmidhuber J "Fast image scanning with deep max-pooling convolutional neural networks". IEEE Int. Conf. Image Process. 2013 https://doi.org/10.48550/arXiv.1302.1700

[12] Guerrero-Higueras, Á. M., Álvarez-Aparicio, C., Calvo Olivera, M. C., Rodríguez-Lera, F. J., Fernández-Llamas, C., Rico, F. M., & Matellán, V. (2019). Tracking people in a mobile robot from 2D lidar scans using full convolutional neural networks for security in cluttered environments. Frontiers in Neurorobotics, 13. https://doi.org/10.3389/fnbot.2018.00085

[13] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. https://doi.org/10.48550/arXiv.1804.02767

[14] M. Marian, F. Stîngă, M. -T. Georgescu, H. Roibu, D. Popescu and F. Manta, "A ROS-based Control Application for a Robotic Platform Using the Gazebo 3D Simulator," 2020 21th International Carpathian Control Conference (ICCC), 2020, pp. 1-5, https://doi.org/10.1109/ICCC49264.2020.9257256

[15] S. Gobhinath, K. Anandapoorani, K. Anitha, D. D. Sri and R. DivyaDharshini, "Simultaneous Localization and Mapping [SLAM] of Robotic Operating System for Mobile Robots," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), 2021, pp. 577-580, https://doi.org/10.1109/ICACCS51430.2021.9441758

[16] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, https://doi.org/10.1109/CVPR.2016.91

[17] Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. (2022). A Review of Yolo Algorithm Developments. Procedia Computer Science, 199, 1066–1073. https://doi.org/10.1016/j.procs.2022.01.135

[18] X. Zhang, L. Zhang and D. Li, "Transmission Line Abnormal Target Detection Based on Machine Learning Yolo V3," 2019 International Conference on Advanced Mechatronic Systems (ICAMechS), 2019, pp. 344-348, https://doi.org/10.1109/ICAMechS.2019.8861617

[19] Bhuiyan MR, Abdullah J, Hashim N, Al Farid F, Ahsanul Haque M, Uddin J, Mohd Isa WN, Husen MN, Abdullah N. 2022. A deep crowd density classification model for Hajj pilgrimage using fully convolutional neural network. PeerJ Computer Science 8:e895 https://doi.org/10.7717/peerj-cs.895

[20] Li, L., Jia, Z., Cheng, T., & Jia, X. (2011). Optimal model predictive control for path tracking of autonomous vehicle. Proceedings - 3rd International Conference on Measuring Technology and Mechatronics Automation, ICMTMA 2011, 2, 791–794. https://doi.org/10.1109/ICMTMA.2011.481

[21] Tavernini, D., Metzler, M., Gruber, P., & Sorniotti, A. (2019). Explicit Nonlinear Model Predictive Control for Electric Vehicle Traction Control. IEEE Transactions on Control Systems Technology, 27(4), 1438–1451. https://doi.org/10.1109/TCST.2018.2837097

[22] Maddalena, E. T., da Moraes, C. G. S., Waltrich, G., & Jones, C. N. (2020). A neural network architecture to learn explicit MPC controllers from data. IFAC-PapersOnLine, 53(2), 11362–11367. https://doi.org/10.1016/j.ifacol.2020.12.546

[23] Farag, K. K. A., Shehata, H. H., & El-Batsh, H. M. (2021). Mobile robot obstacle avoidance based on neural network with a standardization technique. Journal of Robotics, 2021. https://doi.org/10.1155/2021/1129872

[24] Duan, L., Ren, Y., & Duan, F. (2022). Adaptive stochastic resonance based convolutional neural network for image classification. Chaos, Solitons & Fractals, 162, 112429. https://doi.org/10.1016/j.chaos.2022.112429

[25] M. Ahmadi, Z. Xu, X. Wang, L. Wang, M. Shao and Y. Yu, "Fast Multi Object Detection and Counting by YOLO V3," 2021 China Automation Congress (CAC), 2021, pp. 7401-7404, https://doi.org/10.1109/CAC53003.2021.9727949

[26] G. Oltean, C. Florea, R. Orghidan and V. Oltean, "Towards Real Time Vehicle Counting using YOLO-Tiny and Fast Motion Estimation," 2019 IEEE 25th International Symposium for Design and Technology in Electronic Packaging (SIITME), 2019, pp. 240-243, https://doi.org/10.1109/SIITME47687.2019.8990708

[27] Medina-Santiago, A., Camas-Anzueto, J. L., Vazquez-Feijoo, J. A., Hernández-De León, H. R., & Mota-Grajales, R. (2014). Neural Control System in Obstacle Avoidance in Mobile Robots Using Ultrasonic Sensors (Vol. 12). https://doi.org/10.1016/S1665-6423(14)71610-4

[28] Jannah, S. W., & Santoso, A. (2022). Nonlinear Model Predictive Control for Longitudinal and Lateral Dynamic of Autonomous Car. 2022 11th Electrical Power, Electronics, Communications, Controls and Informatics Seminar (EECCIS), 145–148. https://doi.org/10.1109/EECCIS54468.2022.9902927